8. How to detect IDOR vulnerabilities?

Detecting **IDOR (Insecure Direct Object Reference)** vulnerabilities requires a combination of manual and automated approaches, as these vulnerabilities are often context-specific and may be missed by automated tools alone. Here's a guide on effective methods for identifying IDOR vulnerabilities:

1. Manual Testing

- **Identify Parameterized URLs and API Endpoints**: Look for URLs or API endpoints with object identifiers, like `user_id=123`, `document_id=456`, or similar. These often indicate direct references that could be vulnerable to IDOR.
- **Manipulate Object Identifiers**: Manually change parameters or object IDs in the URL, POST data, or API requests to see if unauthorized access to another user's data is possible. For instance, if the original URL is `profile?user_id=1`, try `profile?user_id=2` and observe the response.
- **Check Access Control**: After changing the object identifier, examine the response to see if the application correctly enforces access control, such as returning a "403 Forbidden" status or an error page if access is denied.
- **Session Testing**: Try accessing resources with different accounts, such as an admin and a regular user, to determine if access control mechanisms differ across roles.

2. Automated Tools

- **Burp Suite**: Burp Suite's **Intruder** and **Repeater** tools are especially useful for IDOR testing:
  - **Intruder**: Automate ID parameter manipulation by creating a payload list of possible object IDs and observing which responses indicate unauthorized access.
  - **Repeater**: Send requests repeatedly to test various object IDs while analyzing the responses for unauthorized access to resources.
- **OWASP ZAP**: Use the **Forced Browse** feature to automate requests with different identifiers and check for unexpected access.
- **Fuzzing**: Tools like **wfuzz** or **ffuf** can be configured to send requests with modified identifiers to assess if unauthorized data is accessible.
- **API Security Scanners**: Tools like **Postman** and **Insomnia** can help in testing API endpoints by modifying parameters in requests, especially useful in identifying IDOR vulnerabilities in REST APIs.

3. Identify Common Patterns and Indicators

- **Predictable Identifiers**: Systems with sequential, numeric, or easily guessable identifiers (e.g., `123`, `124`, `125`) are more likely to be vulnerable to IDOR. If the identifier structure is predictable, test for other IDs in the range.
- **Lack of Authorization Checks**: A response that provides unauthorized data or allows actions without requiring explicit permissions (e.g., not requiring authentication to view or modify data) is a

sign of potential IDOR.

4. Testing Workflow Steps

- **Multi-Step Forms**: In multi-step workflows (like e-commerce orders or account management), modify object IDs at each step to see if data from other users can be accessed or modified.
- **Use of Multiple Accounts**: Test the same endpoint with multiple user accounts. Start with a lower-privilege account and modify identifiers to simulate access as a higher-privilege user or another regular user's data.
- **Role-Based Testing**: Ensure resources accessible to certain roles (e.g., admin-only) are inaccessible to regular users by manipulating object identifiers.

5. Look for Inconsistent Error Messages and Status Codes

- **Error Messages**: Unusual or verbose error messages might indicate the presence of an IDOR vulnerability. For example, a response indicating "User not authorized" when altering the `user_id` field could imply insufficient access controls.
- **HTTP Status Codes**: If requests to unauthorized resources return `200 OK` instead of `403 Forbidden`, it can indicate that access control checks aren't enforced correctly.

6. Analyze API Responses for Sensitive Data

- **Examine JSON and XML Responses**: In API testing, carefully review JSON or XML responses. If altering an identifier in the request exposes data from another user, the endpoint might be vulnerable to IDOR.
- **Data Leakage**: Look for personal or sensitive data in responses that shouldn't be accessible with the current user's role or permissions.

7. Cross-Referencing with Other Vulnerabilities

- **Combine with Authentication Bypass**: Attempt IDOR checks alongside other vulnerabilities, like broken authentication or session management issues, to see if combining flaws can increase access.
- **Chain with Other Attacks**: Sometimes, IDOR vulnerabilities are more exploitable when combined with attacks like Cross-Site Scripting (XSS) or Cross-Site Request Forgery (CSRF). This chaining can make it easier to exploit IDOR flaws remotely.

Example Syntax for Manual IDOR Testing (Using Curl)

```
# Replace <TARGET_URL>, <ID_VALUE>, <SESSION_TOKEN> with actual values.
curl -X GET "<TARGET_URL>?user_id=<ID_VALUE>" -H "Authorization: Bearer <SESSION_TOKEN>"
```

This command sends a request to the target URL, modifying the `user_id` parameter to check if an IDOR vulnerability allows unauthorized access to other user data.

Summary of IDOR Detection Steps

Method	Description
Manual Testing	Alter parameters to test access to unauthorized data
Automated Tools	Use tools like Burp Suite Intruder and OWASP ZAP for ID fuzzing
Predictable IDs	Look for sequential or guessable IDs, modify, and observe responses
Workflow Steps	Test multi-step processes for vulnerabilities
Error/Status Codes	Note unusual or inconsistent error messages and HTTP status codes
API Responses	Analyze JSON/XML responses for unexpected data leakage
Vulnerability Chaining	Check if IDOR can be combined with other flaws for stronger impact

Identifying IDOR vulnerabilities often requires an understanding of the application’s logic and user access levels, so combining manual and automated techniques gives the most thorough coverage.